An introduction to Postgres B-Tree indexes

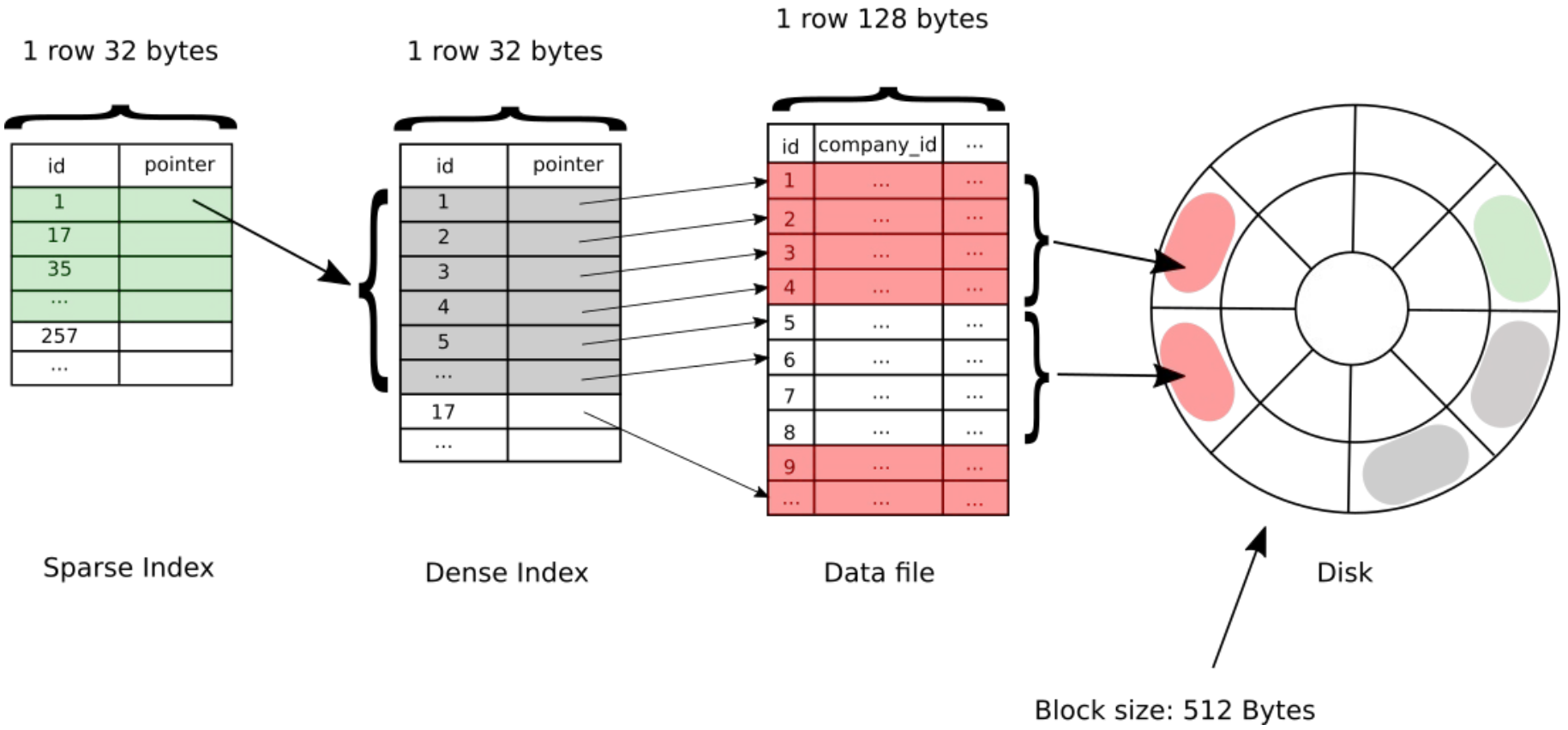
Sebastian Brestin – Data Engineer @ qwertee.io

20 February 2019

Content

- The Index Concept
- B-tree Index Structure
- PotsgreSQL Predicates
- Scans
- Index obfuscation

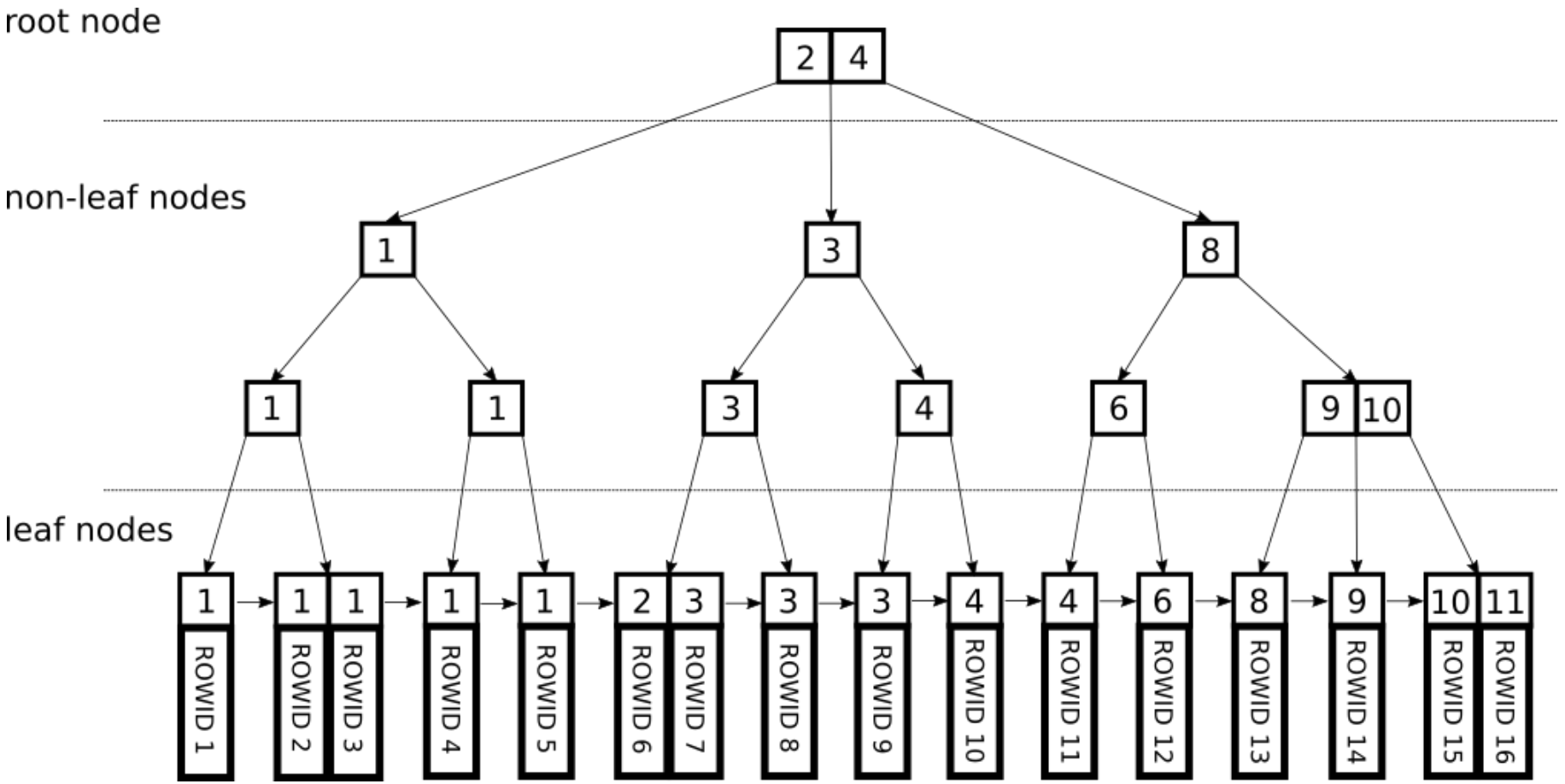
The Index Concept



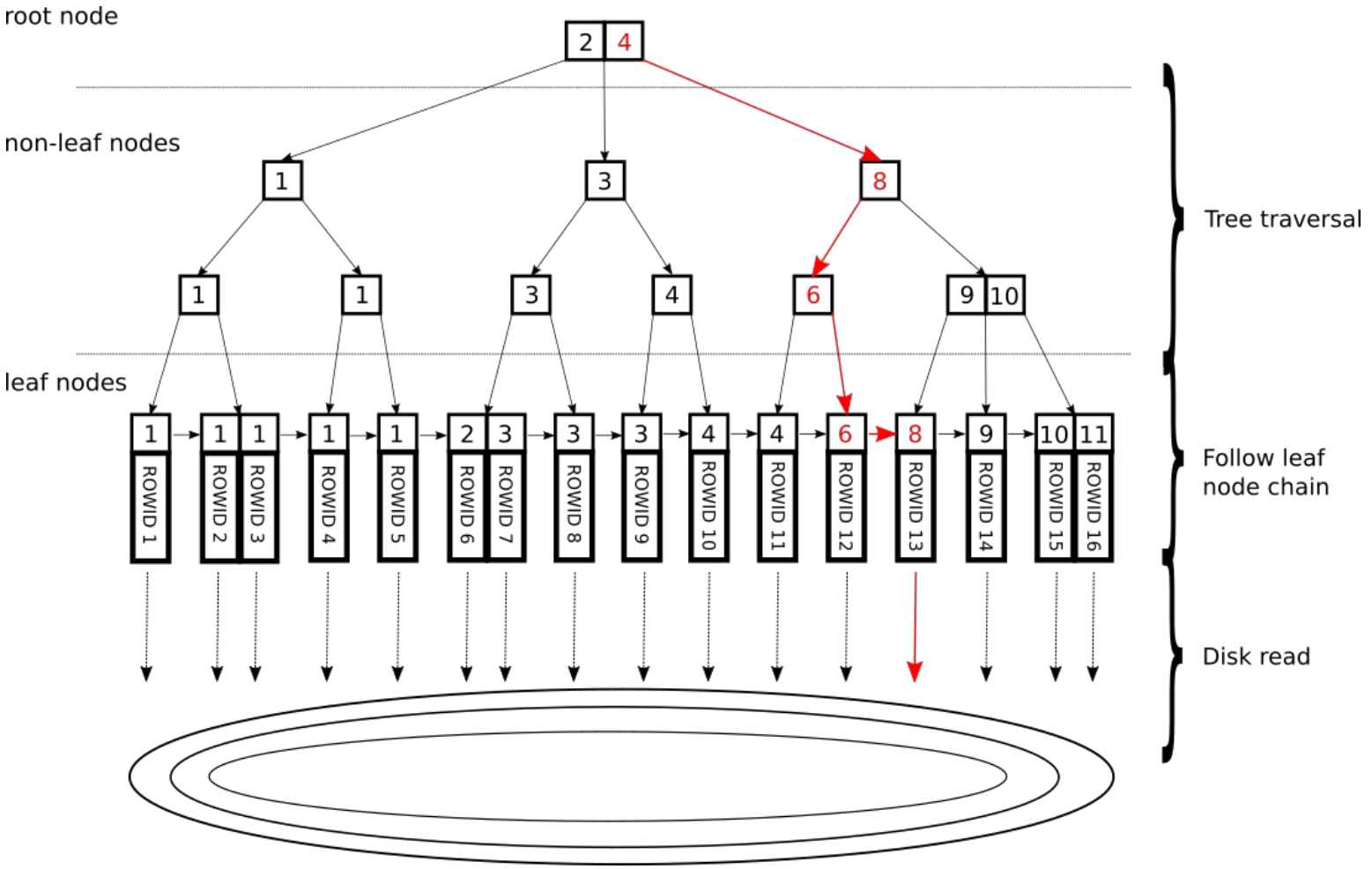
Take away

- Limits disk IO
- A separate data structure
- Storage-time trade-off

B-tree Index Structure



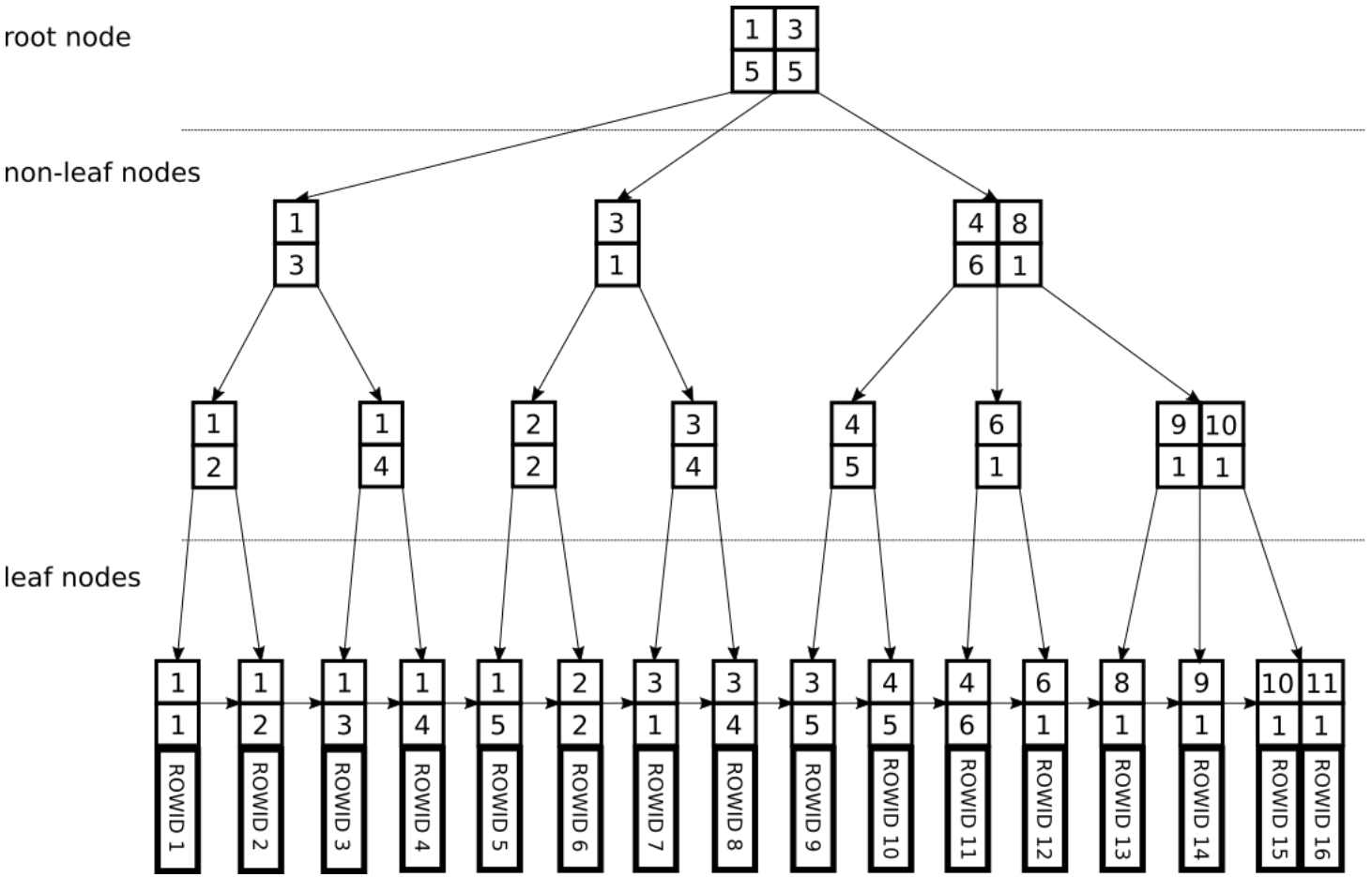
Index Lookup



B-tree Index Types

- Single-column index - an index defined on a single table column
- Multi-column index - an index defined on multiple table columns
- Partial index - an index defined on both columns and rows
- Unique index - an index which enforces the keys in the tree to be unique

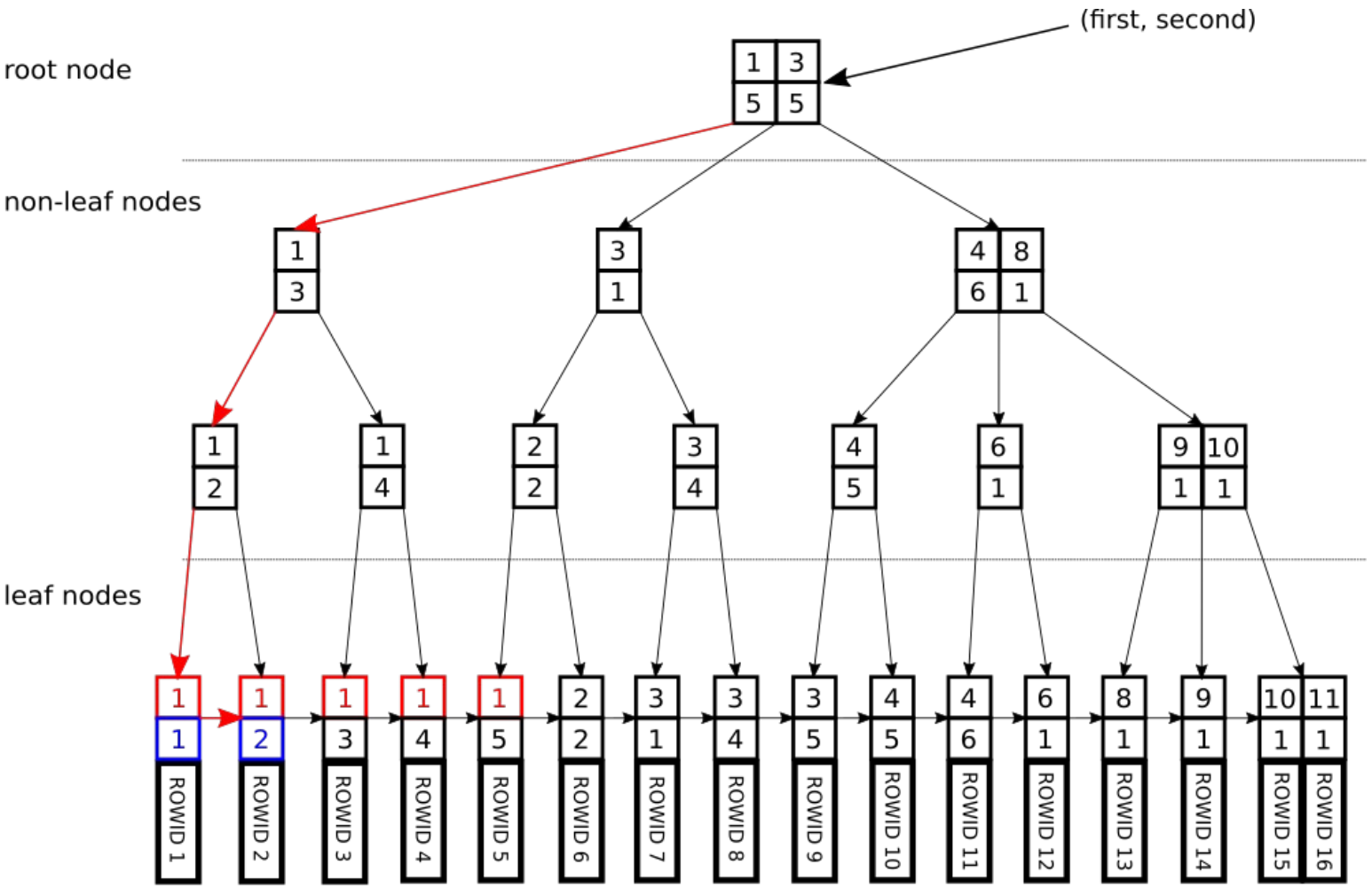
Multi Column Index



Predicates

- Predicates = criteria/condition which follows the WHERE clause
- Postgres supports 3 types of predicates:
  - Index Access Predicate
  - Index Filter Predicate
  - Filter Predicate

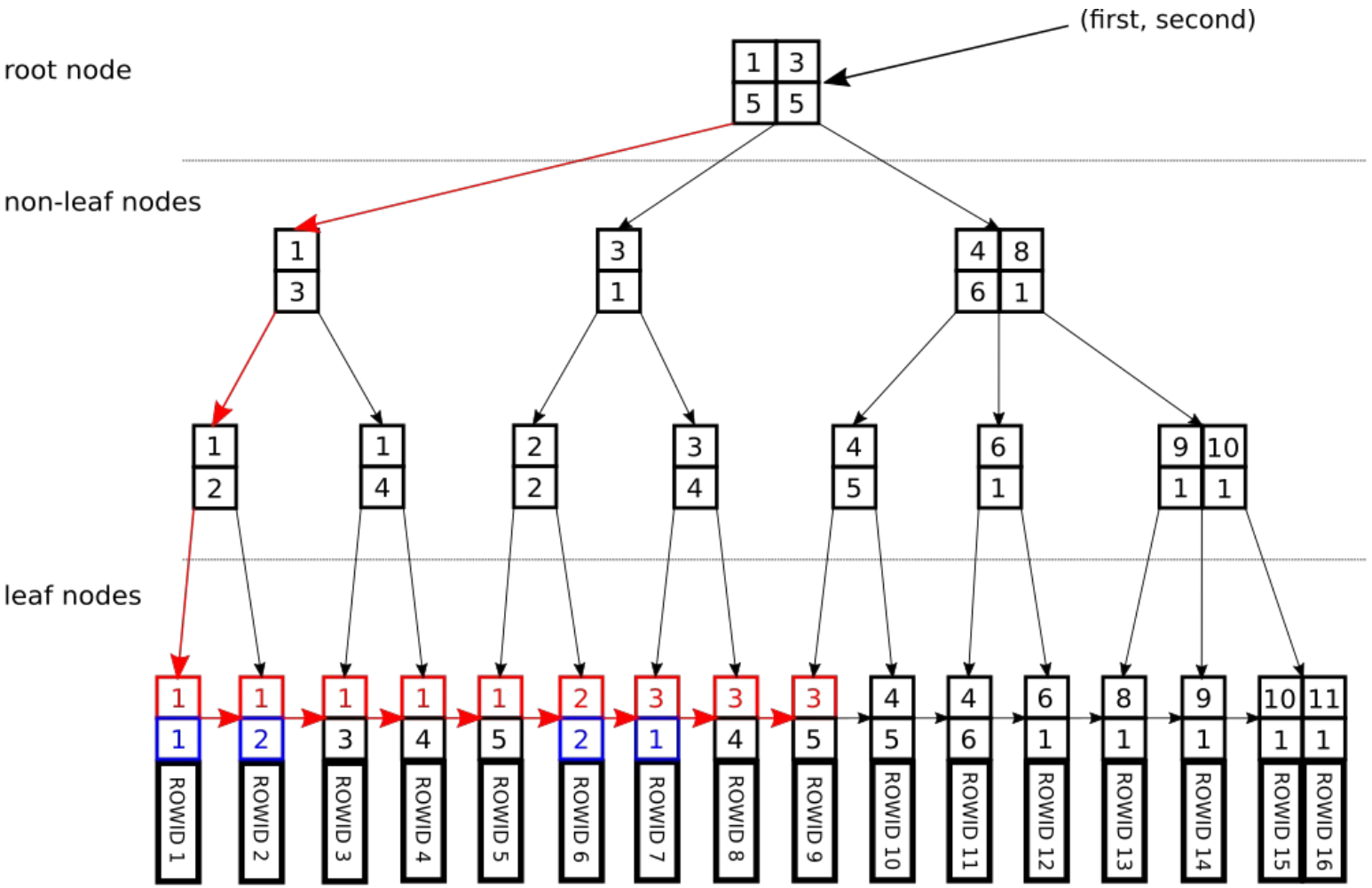
Index Access Predicates



Index Access Predicate: WHERE first=1 AND second>=1 AND second<= 2

Traverse leaf node chain with no interruption

Index Filter Predicates



Index Filter Predicate: WHERE first>=1 AND first<=3 AND second>=1 AND second<= 2

Traverse leaf node chain with jumps

Rule of thumb

- Less granular column first
- Equality first, range after
- Small scan range

Real example

- employees table: (company_id, dep, last_name)
- 100 000 employees
- employees are split equally in 20 departments
- employees are split equally in 100 companies
- grouping by dep = 20 groups of 5000 employees
- grouping by company_id = 100 groups of 1000 employees

why (company_id, dep, last_name) ???

Explanation

- The way the database will be queried:
  - Request all employees from a single department of multiple companies
  - Request all employees from multiple departments of a single company

Rule of thumb

- Always think about the access path when defining an index.

Scans

1. Index Only Scan
2. Index Scan
3. Sequential Scan
4. Bitmap Index Scan

Database table

```
CREATE TABLE "public"."employees" (
    "id" integer NOT NULL,
    "company_id" integer NOT NULL,
    "dep" integer NOT NULL,
    "first_name" character varying(20),
    "last_name" character varying(20),
    "salary" integer,
    "address_id" integer
);
```

Create Index

```
ALTER TABLE employees ADD PRIMARY KEY (id);
```

```
CREATE INDEX ON employees (company_id, dep, last_name);
```

Index Only Scan

Part of the index definition - not disk access

```
EXPLAIN ANALYZE SELECT last_name FROM employees WHERE company_id = 1 AND dep = 10 AND last_name >= 'AF' AND last_name < 'B';
                                                                       QUERY PLAN
--------------------------------------------------------------------------------------------------------------------------------------------------------
 Index Only Scan using employees_company_id_dep_last_name_idx on employees  (cost=0.42..4.44 rows=1 width=8) (actual time=0.034..0.036 rows=2 loops=1)
   Index Cond: ((company_id = 1) AND (dep = 10) AND (last_name >= 'AF'::text) AND (last_name < 'B'::text))
   Heap Fetches: 0
 Planning time: 0.304 ms
 Execution time: 0.078 ms
(5 rows)
```

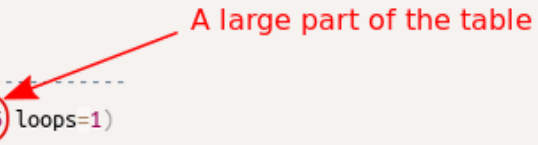
Index Scan

Not part of the index definition - disk access

```
EXPLAIN ANALYZE SELECT first_name FROM employees WHERE company_id = 1 AND dep = 10 AND last_name >= 'AF' AND last_name < 'B';
                                                                    QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------------
 Index Scan using employees_company_id_dep_last_name_idx on employees  (cost=0.42..8.44 rows=1 width=8) (actual time=0.045..0.051 rows=2 loops=1)
   Index Cond: ((company_id = 1) AND (dep = 10) AND ((last_name)::text >= 'AF'::text) AND ((last_name)::text < 'B'::text))
 Planning time: 0.385 ms
 Execution time: 0.103 ms
(4 rows)
```

Sequential Scan

```
EXPLAIN ANALYZE SELECT first_name FROM employees WHERE company_id > 1 AND company_id < 50;
                                                  QUERY PLAN
-------------------------------------------------------------------------------------------------------------------
 Seq Scan on employees  (cost=0.00..2336.00 rows=48423 width=8) (actual time=0.018..13.985 rows=48226 loops=1)
   Filter: ((company_id > 1) AND (company_id < 50))
   Rows Removed by Filter: 51774
 Planning time: 0.177 ms
 Execution time: 15.826 ms
(5 rows)
```



Bitmap Index Scan

```
EXPLAIN ANALYZE SELECT first_name FROM employees WHERE company_id = 1 or company_id = 2;
                                                                          QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------------------------
 Bitmap Heap Scan on employees  (cost=56.43..921.68 rows=1940 width=8) (actual time=0.712..2.988 rows=2006 loops=1)
   Recheck Cond: ((company_id = 1) OR (company_id = 2))
   Heap Blocks: exact=767
   ->  BitmapOr  (cost=56.43..56.43 rows=1950 width=0) (actual time=0.456..0.456 rows=0 loops=1)
         ->  Bitmap Index Scan on employees_company_id_dep_last_name_idx  (cost=0.00..27.64 rows=963 width=0) (actual time=0.279..0.279 rows=979 loops=1)
               Index Cond: (company_id = 1)
         ->  Bitmap Index Scan on employees_company_id_dep_last_name_idx  (cost=0.00..27.82 rows=987 width=0) (actual time=0.176..0.176 rows=1027 loops=1)
               Index Cond: (company_id = 2)
 Planning time: 0.174 ms
 Execution time: 3.256 ms
(10 rows)
```

Bitmap Index Scan

```
EXPLAIN ANALYZE SELECT first_name FROM employees WHERE company_id = 1 or company_id = 2;
                                                   QUERY PLAN
------------------------------------------------------------------------------------------------------
 Bitmap Heap Scan on employees  (cost=56.43..921.68 rows=1940 width=8) (actual time=0.712..2.988 rows=2006 loops=1)
   Recheck Cond: ((company_id = 1) OR (company_id = 2))
   Heap Blocks: exact=767
   ->  BitmapOr  (cost=56.43..56.43 rows=1950 width=0) (actual time=0.456..0.456 rows=0 loops=1)
         ->  Bitmap Index Scan on employees_company_id_dep_last_name_idx  (cost=0.00..27.64 rows=963 width=0) (actual time=0.279..0.279 rows=979 loops=1)
               Index Cond: (company_id = 1)
         ->  Bitmap Index Scan on employees_company_id_dep_last_name_idx  (cost=0.00..27.82 rows=987 width=0) (actual time=0.176..0.176 rows=1027 loops=1)
               Index Cond: (company_id = 2)
 Planning time: 0.174 ms
 Execution time: 3.256 ms
(10 rows)
```

Build 2 bitmap structures

Bitmap Index Scan

```
EXPLAIN ANALYZE SELECT first_name FROM employees WHERE company_id = 1 or company_id = 2;
                                                   QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------
 Bitmap Heap Scan on employees  (cost=56.43..921.68 rows=1940 width=8) (actual time=0.712..2.988 rows=2006 loops=1)
   Recheck Cond: ((company_id = 1) OR (company_id = 2))
   Heap Blocks: exact=767
   ->  BitmapOr  (cost=56.43..56.43 rows=1950 width=0) (actual time=0.456..0.456 rows=0 loops=1)
         ->  Bitmap Index Scan on employees_company_id_dep_last_name_idx  (cost=0.00..27.64 rows=963 width=0) (actual time=0.279..0.279 rows=979 loops=1)
               Index Cond: (company_id = 1)
         ->  Bitmap Index Scan on employees_company_id_dep_last_name_idx  (cost=0.00..27.82 rows=987 width=0) (actual time=0.176..0.176 rows=1027 loops=1)
               Index Cond: (company_id = 2)
 Planning time: 0.174 ms
 Execution time: 3.256 ms
(10 rows)
```

OR bitmap structures

Bitmap Index Scan

```
EXPLAIN ANALYZE SELECT first_name FROM employees WHERE company_id = 1 or company_id = 2;
                                                                           QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------------------------------
 Bitmap Heap Scan on employees  (cost=56.43..921.68 rows=1940 width=8) (actual time=0.712..2.988 rows=2006 loops=1)
   Recheck Cond: ((company_id = 1) OR (company_id = 2))
   Heap Blocks: exact=767
   ->  BitmapOr  (cost=56.43..56.43 rows=1950 width=0) (actual time=0.456..0.456 rows=0 loops=1)
         ->  Bitmap Index Scan on employees_company_id_dep_last_name_idx  (cost=0.00..27.64 rows=963 width=0) (actual time=0.279..0.279 rows=979 loops=1)
               Index Cond: (company_id = 1)
         ->  Bitmap Index Scan on employees_company_id_dep_last_name_idx  (cost=0.00..27.82 rows=987 width=0) (actual time=0.176..0.176 rows=1027 loops=1)
               Index Cond: (company_id = 2)
 Planning time: 0.174 ms
 Execution time: 3.256 ms
(10 rows)
```

Remove records which do not respect the condition

Index Obfuscation

```
EXPLAIN ANALYZE SELECT first_name FROM employees WHERE company_id = 1 AND dep = 10 AND LOWER(last_name) >= 'af' AND LOWER(last_name) < 'b';
                                                                     QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------------------
 Bitmap Heap Scan on employees  (cost=4.91..163.63 rows=1 width=8) (actual time=0.132..0.400 rows=2 loops=1)
   Recheck Cond: ((company_id = 1) AND (dep = 10))
   Filter: ((lower((last_name)::text) >= 'af'::text) AND (lower((last_name)::text) < 'B'::text))
   Rows Removed by Filter: 53
   Heap Blocks: exact=53
   ->  Bitmap Index Scan on employees_company_id_dep_last_name_idx  (cost=0.00..4.91 rows=49 width=0) (actual time=0.042..0.042 rows=55 loops=1)
         Index Cond: ((company_id = 1) AND (dep = 10))
 Planning time: 0.237 ms
 Execution time: 0.461 ms
(9 rows)
```

Applying functions

Index Obfuscation

```
EXPLAIN ANALYZE SELECT first_name FROM employees WHERE id::text = '2';
                                                  QUERY PLAN
-------------------------------------------------------------------------------------------------------------
 Seq Scan on employees  (cost=0.00..2586.00 rows=500 width=8) (actual time=0.022..15.522 rows=1 loops=1)
   Filter: ((id)::text = '2'::text)
   Rows Removed by Filter: 99999
 Planning time: 0.125 ms
 Execution time: 15.553 ms
(5 rows)
```

Casting

Index Obfuscation

```
EXPLAIN ANALYZE SELECT first_name FROM employees WHERE dep > 2;
                                                  QUERY PLAN
-------------------------------------------------------------------------------------------------------------------
 Seq Scan on employees  (cost=0.00..2086.00 rows=90000 width=8) (actual time=0.016..14.855 rows=89840 loops=1)
   Filter: (dep > 2)
   Rows Removed by Filter: 10160
 Planning time: 0.147 ms
 Execution time: 18.105 ms
(5 rows)
```

Not using an index prefix

Index Obfuscation

```
EXPLAIN ANALYZE SELECT first_name FROM employees WHERE id * 100 > 50000;
                                                  QUERY PLAN
-------------------------------------------------------------------------------------------------------------------
 Seq Scan on employees  (cost=0.00..2336.00 rows=33333 width=8) (actual time=0.169..14.959 rows=99500 loops=1)
   Filter: ((id * 100) > 50000)
   Rows Removed by Filter: 500
 Planning time: 0.122 ms
 Execution time: 18.352 ms
(5 rows)
```

Performing arithmetics

Conclusion

- Indexes are meant to reduce IO:
  - Think about the way the data is queried
  - Think about how much data you request

More

- https://www.qwertee.io/blog/postgresql-b-tree-index-explained-part-1/
- https://github.com/sebrestin/postgres-btree-dataset
- https://github.com/sebrestin

Thank you